CSE 259 Fall 2019

Project 3

online

1. Project description

In this project, you will apply logic programming to write your first theorem prover! You will gain additional hands-on experience with logic programming and insights into how computers may be used to solve logic problems and prove theorems. This is an individual project. Happy prolog!

2. Requirement specification:

In this project, you will write a theorem prove using Wang's algorithm with GProlog.

1. We are now ready to work on your first theorem prover. For this, we are going to use Wang's algorithm, which is described as follows:
2. Rules to determine the validity of a sequent:

   a. Write the premises down separated by commas, followed by an arrow and then the theorem to be proved. E.g.:

      p, ~q ^ r ⇒ p ^ ~q [~ denotes negation, ^ conjunction, v disjunction, ⇒ sequent, -> implication]

      The aim is now to progressively remove connectives from this.

   b. If one of the formulae separated by commas is the negation of a formula, drop the negation sign and move it to the other side of the arrow. E.g.:

      p, ~(q ^ r) ⇒ p ^ r

      is changed to:

      p ⇒ p ^ r, q ^ r

   c. If the last connective of a formula on the left is ^ (and), or on the right of the arrow is v (or), replace the connective by a comma. E.g.:

      p, p ^ q ⇒ r, s

      is changed to:

      p, p, q ⇒ r, s

      and

      p, q ⇒ r v p

is changed to:

p, q ⇒ r, p

d. If the last connective of a formula on the left is v (or), or on the right of the arrow is ^ (and), then produce two new lines, each with one of the two sub-formulae replacing the formula. Both of these must be proved in order to prove the original theorem. E.g.:

p v q, r, s ⇒ q v p

is changed to the two lines:

p, r, s ⇒ q v p
q, r, s ⇒ q v p

and, similarly, p, r, s ⇒ q ^ p is changed to two lines:

p, r, s ⇒ q
p, r, s ⇒ p

Each of these must be independently and successfully dealt with in order to succeed with the proof.

e. If the last connective of a formula on the right is A → B, remove A → B from the right and then add A to the left and B to the right. E.g.:

p v q ⇒ q → p

is changed to:

p v q, q ⇒ p

f. If the last connective of a formula on the left is A → B, remove A → B from the left and then create two new lines, one with B added to the left, and the other with A added to the right. E.g.:

p, p → q ⇒ p v q

is changed to:

p, q ⇒ p v q
p ⇒ p v q, p

g. If the same atom or formula ever appears on both sides of an arrow, the line is proved.

h. If no connectives (~, ^, v, or →) remain in a line, and no atom appears on both sides of the arrow, the line cannot be proved.

3. Find a file called "**thmpv.pl**" in the download file, with the theorem prover already partially implemented for you. What you need to write is some of the remaining rules. Your task is to finish the remaining rules.

   The places where your inputs are needed are clearly marked. Do not change code in other places or you may lose points.

4. Test and play with your new theorem prover!

5. If you encounter any memory problems, which are unlikely to happen unless there are issues with your code or you are testing very challenging problems. If you are convinced that your code is correct, the next thing to try is changing the memory settings.

3. Submission:

Submission is ***electronically*** via canvas in a zip file. ***One and only one*** member must submit the file. It should contain the following files:

a. README.txt: this file should include names of you team members ***and*** each of your contributions; be precise

b. thmpv.pl: your code; make sure to test it thoroughly and ***comment*** properly for the part you wrote.

4. Grading:

Grading will be based on the following criteria:

a. Implement all remaining rules: 90%

d. Comment: 10%

Note that your contribution to the team ***will be factored*** into your project grade so be active!